Research and Implementation of SemanticSegmentation Network Accelerator Based on Vitis AI

Grace Zhang * and Callahan Rhodes

Posted Date: 17 November 2025

doi: 10.20944/preprints202511.1182.v1

Keywords: on-site programmable gate array; deep learning processing unit; semantic segmentation; vitis AI; convolutional neural network

 doi:10.20944/preprints202511.1182.v1

Article

Research and Implementation of Semantic Segmentation Network Accelerator Based on Vitis AI

Grace Zhang * and Callahan Rhodes

Department of Computer Science, University of Vermont, Burlington, USA
* Correspondence: grace.zhang9970@gmail.com

Abstract

Based on Xilinx Vitis AI, this paper carries out network pruning, deep learning processing unit (DPU) customization, and hardware-software co-optimization methods for the semantic segmentation network U-Net. Finally, the semantic segmentation accelerator is designed and implemented on the Xilinx ZCU102 development platform. Under the premise of relatively low accuracy loss, the consumption of hardware resources is reduced, and the complete software and hardware system development of the U-Net network is completed. Experimental results show that the processing frame rate of the U-Net network hardware accelerator can reach 42 fps, verifying the effectiveness of the acceleration scheme for the neural network.

Keywords: on-site programmable gate array; deep learning processing unit; semantic segmentation; vitis AI; convolutional neural network

1. Introduction

Semantic segmentation is a deep learning task that enables pixel-level labeling of images. It identifies pixels with the same label as having similar semantic features. This technique has been widely applied in professional fields such as autonomous driving, geological detection, and assisted medical diagnosis [1].

Traditional image segmentation methods include edge detection and threshold segmentation. These methods can only utilize shallow features of the image (such as texture or color information), and are limited by manually designed features and prior knowledge, making them less robust to environmental changes. As a result, their segmentation performance under complex scenarios is often unsatisfactory [2].

In recent years, deep learning-based image segmentation algorithms have rapidly advanced. Due to their high accuracy and strong robustness to environmental variation, they have quickly become mainstream solutions for image segmentation [3]. However, in practical applications, deep learning often faces problems such as high computational cost and significant power consumption. According to [4], U-Net implemented on the Pynq-Z1 platform using an ARM Cortex-A9 processor at 650 MHz achieved a total running time of 16,635 s. Reference [5,6] describes edge TPU accelerator board completing FCN deployment with an average frame latency of 0.500 s, and SegNet achieving 0.062 s [7,8].

Reference [9] discusses semantic segmentation networks for autonomous driving scenarios, such as MultiNet [10] and DLT-Net [11], which were tested on the NVIDIA GTX TITAN XP GPU platform with measured latencies of 0.116 s and 0.108 s, respectively. It can be seen that the semantic segmentation task needs to meet demanding performance requirements in real-world scenarios. However, GPU-based acceleration still suffers from issues like high computational cost, thermal management challenges, and large energy consumption. Deep learning models are often computationally intensive and time-consuming. When using CPUs or GPUs, high power consumption and limited portability are common problems.

 doi:10.20944/preprints202511.1182.v1

FPGA, due to its flexible configurability and low power consumption, is a potential alternative to CPUs and GPUs. The custom hardware design of neural network accelerators based on FPGA has developed rapidly in recent years [12].

This paper proposes a method based on hardware-software co-design on a single SoC to accelerate the semantic segmentation network U-Net on FPGA. It improves latency and energy efficiency, performs network quantization, and evaluates the hardware architecture based on DPU. A kind of embedded system-based high-performance, low-power semantic segmentation network accelerator on FPGA is designed and implemented.

2. Related Work

Semantic segmentation has been a central task in computer vision, with continued advancements in network architectures that improve precision and computational efficiency. Transformer-CNN fusion networks have shown effectiveness in tasks like medical image segmentation, advancing the robustness of pixel-level prediction under complex conditions [13].

Alongside vision models, large-scale language models have spurred innovations in retrieval-augmented generation (RAG). Several works have introduced selective knowledge injection through adapter modules [14], contrastive alignment for text-to-image generation [15], and intelligent RAG modeling for knowledge fusion [16]. These methods support semantic-rich outputs and contextual understanding, which are transferable to structured inference tasks like segmentation acceleration. Other RAG techniques, such as fusion-based QA systems [17], prompt-based controllable abstraction [18], and retrieval agents for literature summarization [19], demonstrate the role of dynamic context integration in model outputs.

Reinforcement learning (RL) has been widely applied in intelligent scheduling and optimization systems. Applications include multi-agent scheduling in microservices [20], elastic resource scaling in cloud environments [21], and autonomous resource management via RL [22]. In addition, task decomposition and dynamic collaboration frameworks driven by language models [23], as well as adaptive interaction strategies [24], extend RL methodologies into highly modular and scalable infrastructures. Rate-limiting in microservices has also benefited from multi-objective adaptive RL techniques [25].

Anomaly detection remains crucial for system reliability and monitoring. Federated anomaly detection frameworks address privacy and personalization needs in multi-tenant cloud platforms [26], while models using Siamese networks improve discrimination accuracy in financial scenarios [27]. Health risk prediction using federated learning [28] and anomaly detection in ETL pipelines with autoencoders [29] emphasize the importance of secure and decentralized learning approaches. Multiscale temporal modeling techniques further enhance detection sensitivity in cloud services [30].

Graph neural networks (GNNs) and structure-aware learning methods contribute to both interpretability and systemic modeling. Techniques like temporal graph attention for clinical data [31], GNN-based routing strategies in microservices [32], and enterprise-level credit risk identification [33] offer structural insights that complement statistical learning. The integration of knowledge graphs and attention mechanisms in recommendation systems [34] reflects the growing role of graph structures in transparent and explainable AI.

Deep sequence modeling and attention-based time series forecasting have also proven effective in financial fraud discrimination [35] and systemic risk analysis [36]. These techniques, while domain-specific, contribute to the development of generalized architectures that are applicable to performance-critical applications such as hardware-accelerated segmentation systems.

3. Semantic Segmentation Network

3.1. Algorithm Principle

U-Net is a network modified and extended based on the FCN architecture, which can achieve more accurate segmentation with fewer training images. The structure of U-Net, as shown in Figure

 doi:10.20944/preprints202511.1182.v1

1, is a typical encoder-decoder architecture. The symbols in the lower right corner of the figure sequentially represent convolution layers with a kernel size of 3×3 (followed by ReLU activation), cropping and copy layers, max pooling layers with a kernel size of 2×2, and upsampling layers also with a kernel size of 2×2. The entire U-Net network performs 19 convolutions, and 4 times of pooling, upsampling, cropping, and copying operations.

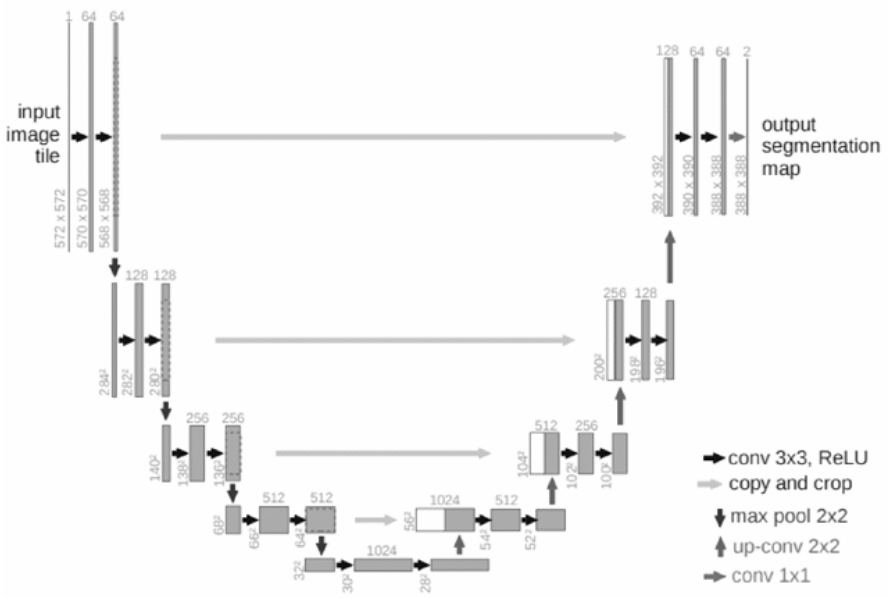

Figure 1. U-Net Architecture.

3.2. Computation Performance Analysis

The U-Net network structure used in this paper consists of 8 layers. On a general-purpose computing platform with a power consumption of 165 W and a main frequency of 3.5 GHz using an i9-9900X processor, a performance analysis of the model was conducted. The results show that processing an image of size 1024×512×3 requires 6.49 s. The high power consumption and long latency fail to meet the demands of many real-time scenarios.

Therefore, this paper reduces computation by network pruning and addresses the issues of large computation and long processing time by hardware acceleration through custom optimization of the DPU architecture.

4. Custom Implementation of Software-Hardware System Based on Vitis AI and DPU

4.1. Vitis AI Design Method

4.1.1. Main Workflow

Vitis AI is used on Xilinx platforms (such as edge devices or FPGA accelerator cards) to deploy and accelerate deep neural networks. The Vitis AI stack includes various functions such as model compression, computational optimization, and architecture optimization, enabling efficient acceleration of deep learning models mapped to FPGAs. Model compression optimization includes parameter pruning and data quantization, aiming to minimize accuracy loss while improving efficiency.

The development process of Vitis AI requires Vitis AI and Vitis IDE. As shown in Figure 2, the Vitis AI workflow includes three basic steps. First, on the host computer, a model is built using Vitis AI with pre-trained floating-point models as input. Then, the Vitis software platform is used to build a customized hardware platform, generating hardware including DPU IP and other logic cores. Finally, the compiled executable software is run on the constructed hardware, using C++ to call Vitis AI for model loading and execution.

 doi:10.20944/preprints202511.1182.v1

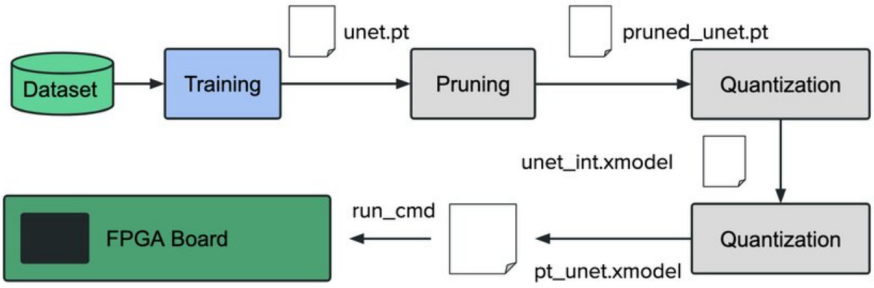

Figure 2. Vitis AI Workflow Diagram.

4.1.2. DPU Architecture

The deep learning processing unit (DPU) is a general-purpose programmable accelerator designed to accelerate deep learning model inference. The DPU supports a large number of basic operations used in deep learning, such as convolution, deconvolution, max pooling, and fully connected layers. Figure 3 shows the hardware architecture of the DPU. The DPU parameters can be configured according to application needs to optimize resource usage and utilization, enabling selection of the required functions.

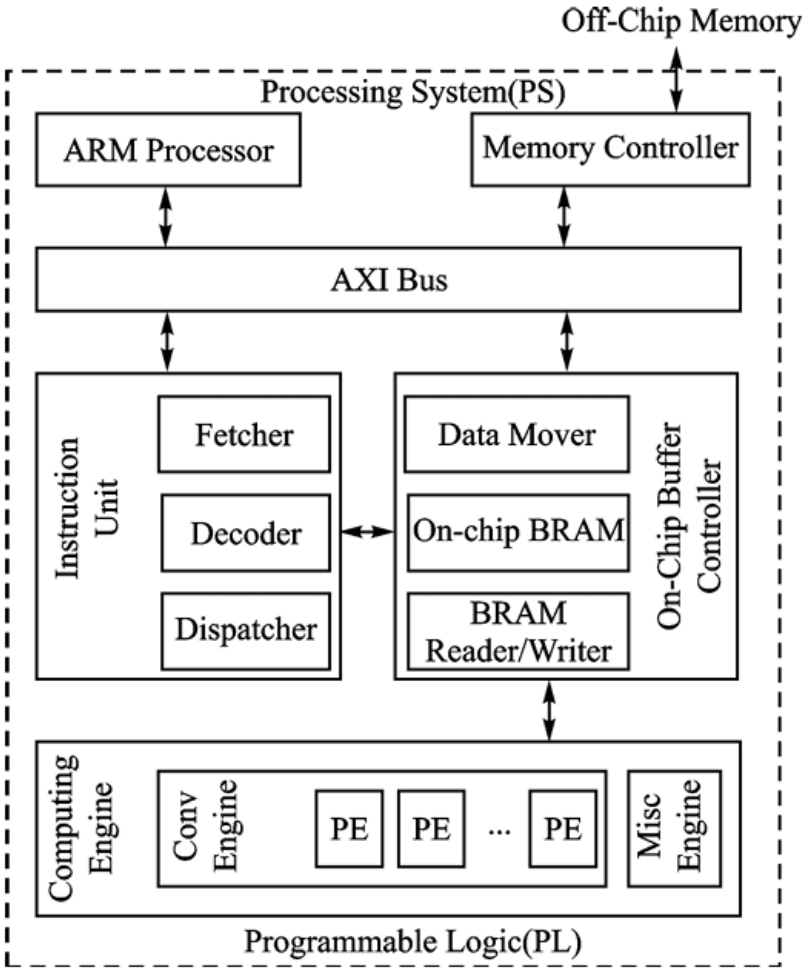

Figure 3. DPU Architecture.

4.2. Network Quantization

Network inference is a computation-intensive process that requires large memory bandwidth to meet the low-latency and high-throughput demands of edge applications. Since the DPU is implemented on FPGA, and the DSP units in FPGAs offer better support for fixed-point operations

 doi:10.20944/preprints202511.1182.v1

[13], in Vitis AI, the AI Quantizer is used to convert FP32 models into INT8 or even lower precision models for FPGA deployment.

As shown in Figure 4, the quantization workflow takes a pre-trained floating-point model as input. The main tasks of the preprocessing stage are to fold and remove redundant nodes, then quantize the model's weights, biases, and activations into fixed-point bit widths. To improve the final model's accuracy, the quantization process requires image data inputs to run multiple inference iterations for calibration. Calibration helps transform the quantized model into a deployable DPU model.

In the Vitis AI workflow, the AI Quantizer is an essential component. Because FPGA supports fixed-point computing well, the Quantizer converts 32-bit floating-point operations into 8-bit fixed-point operations, minimizing the accuracy loss during quantization. In this paper, FP32 operations in the network are converted to INT8 operations during quantization. Under the premise of tolerable prediction accuracy loss, the computational complexity of the model is significantly reduced, and both inference speed and efficiency are improved.

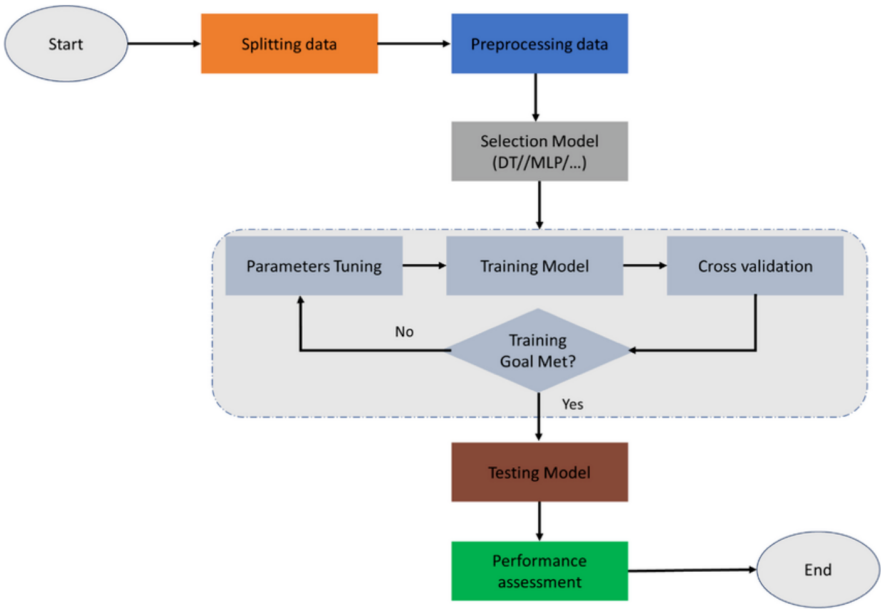

Figure 4. Quantization Workflow Diagram.

4.3. Improving Computational Performance Through DPU Configuration

The DPU compiler converts neural network models into a series of DPU instructions, triggering the DPU to fetch and execute them from off-chip memory via a programmable control logic unit. The computing engine adopts a deeply pipelined architecture and includes one or more processing elements (PEs). Each PE consists of multipliers, adders, and accumulators.

The DSP module operates at a clock frequency configured to be twice that of general logic. The DPU stores model parameters and intermediate features in on-chip BRAM to buffer input/output and intermediate data, reducing external memory bandwidth consumption. The DPU communicates with the processing system (PS) via the AXI bus, which includes a low-bandwidth interface for command fetching and two high-bandwidth interfaces for data transfer.

Table 1 shows the configurable parameters of the DPU. The architecture indicates the DPU's achievable peak operations and clock frequency. For example, the smallest B512 architecture operates at a peak frequency of 512 MHz, while the largest B4096 architecture can reach 4096 MHz. Based on the available board resources and the target network, this paper adopts the B4096 architecture for DPU IP implementation.

RAM usage refers to storing neural network weights, bias parameters, and intermediate features in on-chip BRAM. DSP usage indicates the resource balance during PE operations (whether more DSPs or LUTs are used). In this design, all configurations adopt high utilization.

 doi:10.20944/preprints202511.1182.v1

The low power mode disables the DPU clock during idle periods to save power. In this experiment, the mode is enabled. The extra operations list the optional functions supported by the DPU, which can be disabled to conserve hardware resources.

Table 1. DPU Configuration Information.

Parameter	Meaning	Configuration
Arch of DPU	DPU Architecture	B4096, B3136, B2304, B1600, B1152, B1024, B800, B512
Number of DPUs	Number of DPUs	1, 2, 3, 4
RAM Usage	RAM Utilization	High, Low
DSP Usage	DSP Utilization	High, Low
Low Power Mode	Low Power Mode	Enabled, Disabled
Extra operations	Optional Features	Softmax, Depthwise Convolution, Average-Pool, LeakyReLU, Elementwise Multiplication

4.4. Overall Design

Figure 5 shows the overall hardware accelerator design of the DPU combined with the ARM processor. The semantic segmentation model is managed by a custom API, which uses Vitis AI library functions to handle communication between the APU and the DPU accelerator. The Vitis AI API library communicates with the DPU driver to transfer data between the processor and the accelerator.

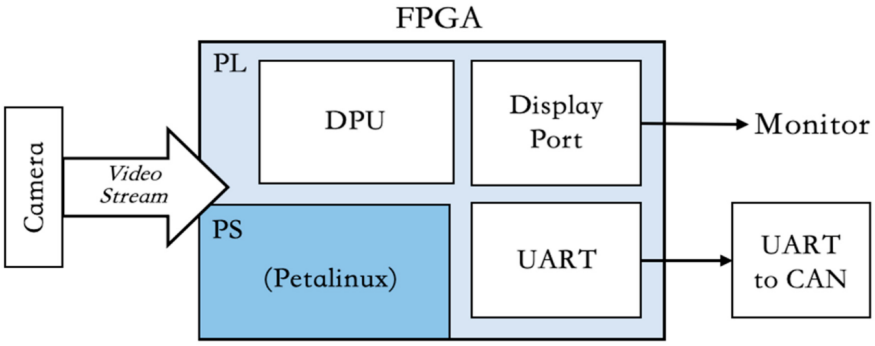

Figure 5. Overall System Design.

As shown in Figure 5, the APU controls the entire system to receive images. The images are classified and segmented by the accelerator, and the inference results are then passed to the APU. At the system level, data are initially stored in off-chip DDR-RAM, which is managed by the DDR controller through the AXI bus.

The system first loads the data from off-chip storage and assigns part of it to the DPU buffer area for acceleration. Once the DPU finishes one round of processing, the data will be written back to the off-chip buffer and then returned to internal memory for post-processing. The transferred data includes input images of size 1024×512, model weights, model deviations, and related model instructions.

During runtime, the DPU fetches the instructions and data related to the model from off-chip memory. These are used to control the operations of the processing elements (PEs) in the DPU's mixed computing array. The data are then loaded into the DPU for accelerated inference.

To reduce total memory bandwidth usage, the system attempts to reuse data as much as possible, thereby improving the overall performance in terms of latency and energy efficiency.

5. Experimental Results and Analysis

5.1. Software and Hardware Environment

FPGA development tools: Xilinx Vitis AI 2.0, Xilinx Vitis 2021.2, Ubuntu 20.04.

 doi:10.20944/preprints202511.1182.v1

FPGA platform: Xilinx Zynq UltraScale+ XCZU9EG-2FFVB1156E MPSoC development board.

5.2. Experimental Results of Network Model Quantization

The Cityscapes dataset was used as the input image source. As training progressed, mIOU was periodically measured. Based on 500 validation images from the Cityscapes validation dataset, the model was evaluated.

As shown in Figure 6, the mIOU changed during the process. After 13,000 iterations, the mIOU reached 0.546. Therefore, this paper selects the floating-point model after 13,000 iterations for quantization and deployment.

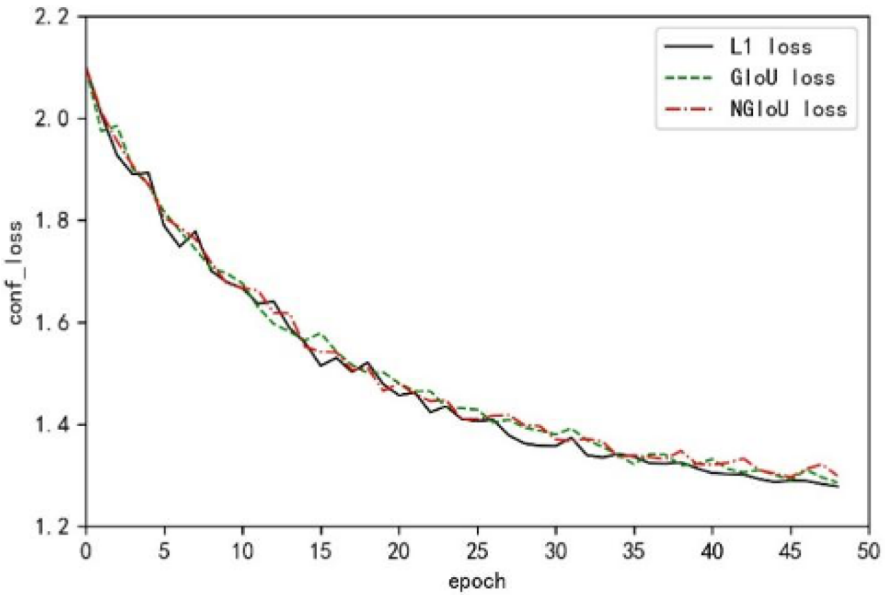

Figure 6. Average mIOU.

After quantization, the neural network model parameters are shown in Table 2. The parameter size is reduced to approximately one-fourth of the original. As seen from the table, with 8-bit quantization, the mIOU shows almost no loss. Under the condition of negligible accuracy reduction, the model achieves significant parameter compression, indicating that the quantization process is effective.

Table 2. Comparison of Model Parameters Before and After Quantization.

Model	mIOU	Parameters	Storage Size / MB
FP32	0.546	31.03M	59.25
INT8	0.539	8.26M	29.61

5.3. Analysis of Experimental Results After DPU Configuration

Table 3 lists the hardware resource usage and utilization after DPU configuration. By analyzing the computational performance of the DPU during operation, it is shown that after optimization, the total inference time accelerated by the DPU is 23.6 ms, with a system segmentation frame rate of 42 fps and power consumption of 21.7 W.

Table 3. DPU Resource Utilization.

Resource	Used	Total	Utilization / %
LUT	79078	274080	28.85
FF	76921	548160	14.03
BRAM	235	912	25.77
DSP	312	2520	12.38

 doi:10.20944/preprints202511.1182.v1

Table 4 shows a comparison of the experimental results between the proposed method and the GPU platform. It can be seen that the design proposed in this paper, compared to the floating-point model on the GPU platform, although has a single image processing time of 23.6 ms, the actual power consumption on the FPGA platform is lower than that of the GPU platform. The overall performance-power ratio is five times higher than that of the GPU platform.

Table 4. Comparison with Other Platforms.

Method	FPGA	Power (W)	Time (ms)	FPS	Perf. (fps·W^{-1})
This Work	Zynq ZCU102	21.7	23.6	42.37	1.95
GPU	RTX 2080 Ti	223	12.3	81.30	0.364

6. Conclusions

This paper takes the U-Net semantic segmentation algorithm as an example and presents the design and implementation of a convolutional neural network accelerator architecture using DPU configuration and quantization. It introduces the complete workflow of deploying convolutional neural network models to FPGAs using Vitis AI.

At the same time, the U-Net network structure from the front end to the back end was modified and processed with Vitis AI, including data quantization and model compression. These operations reduce network scale and minimize access time to off-chip memory for the accelerator.

The semantic segmentation accelerator implemented on FPGA hardware achieved a frame rate of 42 fps. Future work will continue to explore semantic segmentation accelerators based on FPGA, including deploying multiple DPU-based networks for specific real-world applications, aiming to achieve better performance and low-power neural network acceleration.

References

1. J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 3431–3440, 2015.
2. V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.
3. L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.
4. O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, pp. 234–241, 2015.
5. G. Cuni, "Deploying Deep Learning on FPGA: an assessment of ConvNets performance on Xilinx Zynq MPSoC using Vitis-AI development platform," Doctoral dissertation, Politecnico di Torino, 2022.
6. A. Ghaffari, D. Soudbakhsh and S. Jain, "Efficient FPGA-based CNN inference accelerator using HLS," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1–5, 2020.
7. N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, pp. 1–12, 2017.
8. H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 2881–2890, 2017.
9. M. Teichmann, M. Weber, M. Zoellner, R. Cipolla and R. Urtasun, "MultiNet: Real-time joint semantic reasoning for autonomous driving," in *Proc. IEEE Intell. Vehicles Symp.*, pp. 1013–1020, 2018.
10. W. Liu, A. Rabinovich and A. C. Berg, "ParseNet: Looking wider to see better," arXiv:1506.04579, 2015.
11. Y. Yang and R. Urtasun, "DLT-Net: Joint object detection and semantic segmentation for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1219–1228, 2020.
12. C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, pp. 161–170, 2015.
13. X. Wang, X. Zhang and X. Wang, "Deep Skin Lesion Segmentation with Transformer-CNN Fusion: Toward Intelligent Skin Cancer Analysis," arXiv:2508.14509, 2025.

14. H. Zheng, L. Zhu, W. Cui, R. Pan, X. Yan and Y. Xing, "Selective Knowledge Injection via Adapter Modules in Large-Scale Language Models," 2025.
15. D. Gao, "High Fidelity Text to Image Generation with Contrastive Alignment and Structural Guidance," arXiv:2508.10280, 2025.
16. D. Wu and S. Pan, "Joint Modeling of Intelligent Retrieval-Augmented Generation in LLM-Based Knowledge Fusion," 2025.
17. Y. Sun, R. Zhang, R. Meng, L. Lian, H. Wang and X. Quan, "Fusion-based retrieval-augmented generation for complex question answering with LLMs," in *Proc. 2025 8th Int. Conf. Comput. Inf. Sci. Appl. Technol. (CISAT)*, pp. 116–120, 2025.
18. X. Song, Y. Liu, Y. Luan, J. Guo and X. Guo, "Controllable Abstraction in Summary Generation for Large Language Models via Prompt Engineering," arXiv:2510.15436, 2025.
19. J. Zheng, Y. Chen, Z. Zhou, C. Peng, H. Deng and S. Yin, "Information-Constrained Retrieval for Scientific Literature via Large Language Model Agents," 2025.
20. R. Zhang, "AI-driven multi-agent scheduling and service quality optimization in microservice systems," *Trans. Comput. Sci. Methods*, vol. 5, no. 8, 2025.
21. B. Fang and D. Gao, "Collaborative Multi-Agent Reinforcement Learning Approach for Elastic Cloud Resource Scaling," arXiv:2507.00550, 2025.
22. Y. Zou, N. Qi, Y. Deng, Z. Xue, M. Gong and W. Zhang, "Autonomous resource management in microservice systems via reinforcement learning," in *Proc. 2025 8th Int. Conf. Comput. Inf. Sci. Appl. Technol. (CISAT)*, pp. 991–995, 2025.
23. S. Pan and D. Wu, "Modular Task Decomposition and Dynamic Collaboration in Multi-Agent Systems Driven by Large Language Models," arXiv:2511.01149, 2025.
24. R. Liu, Y. Zhuang and R. Zhang, "Adaptive Human-Computer Interaction Strategies Through Reinforcement Learning in Complex," arXiv:2510.27058, 2025.
25. N. Lyu, Y. Wang, Z. Cheng, Q. Zhang and F. Chen, "Multi-Objective Adaptive Rate Limiting in Microservices Using Deep Reinforcement Learning," arXiv:2511.03279, 2025.
26. Y. Wang, H. Liu, N. Long and G. Yao, "Federated anomaly detection for multi-tenant cloud platforms with personalized modeling," in *Proc. 2025 5th Int. Conf. Intell. Commun. Comput. (ICICC)*, pp. 555–559, 2025.
27. H. Feng, Y. Wang, R. Fang, A. Xie and Y. Wang, "Federated Risk Discrimination with Siamese Networks for Financial Transaction Anomaly Detection," 2025.
28. R. Hao, W. C. Chang, J. Hu and M. Gao, "Federated Learning-Driven Health Risk Prediction on Electronic Health Records Under Privacy Constraints," 2025.
29. X. Chen, S. U. Gadgil, K. Gao, Y. Hu and C. Nie, "Deep Learning Approach to Anomaly Detection in Enterprise ETL Processes with Autoencoders," arXiv:2511.00462, 2025.
30. L. Lian, Y. Li, S. Han, R. Meng, S. Wang and M. Wang, "Artificial intelligence-based multiscale temporal modeling for anomaly detection in cloud services," arXiv:2508.14503, 2025.
31. X. Zhang and Q. Wang, "EEG Anomaly Detection Using Temporal Graph Attention for Clinical Applications," *J. Comput. Technol. Softw.*, vol. 4, no. 7, 2025.
32. C. Hu, Z. Cheng, D. Wu, Y. Wang, F. Liu and Z. Qiu, "Structural Generalization for Microservice Routing Using Graph Neural Networks," arXiv:2510.15210, 2025.
33. Y. Lin, "Graph neural network framework for default risk identification in enterprise credit relationship networks," *Trans. Comput. Sci. Methods*, vol. 4, no. 4, 2024.
34. S. Lyu, M. Wang, H. Zhang, J. Zheng, J. Lin and X. Sun, "Integrating Structure-Aware Attention and Knowledge Graphs in Explainable Recommendation Systems," arXiv:2510.10109, 2025.
35. Z. Xu, J. Xia, Y. Yi, M. Chang and Z. Liu, "Discrimination of Financial Fraud in Transaction Data via Improved Mamba-Based Sequence Modeling," 2025.
36. Q. R. Xu, W. Xu, X. Su, K. Ma, W. Sun and Y. Qin, "Enhancing Systemic Risk Forecasting with Deep Attention Models in Financial Time Series," 2025.